University of Newcastle
Discipline of Computer Science and Software Engineering
Semester 2, 2016 - SENG1120/6120

Assignment 2

Due using the Blackboard Assignment submission facility:
11:59PM – Friday, 4 November 2016

NOTE: *The important information about submission and code specifics at the end of this assignment specification***.**

In lectures we have discussed the use of binary trees. This assignment will require the implementation of a binary tree and several algorithms for element insertion, removal, search and tree traversal.

Assignment Task

Your task in this Assignment is to implement a data structure based on a binary search tree, which will store generic objects, i.e. use templates.

The type of object used in this assignment is called `student` and it should store two data values: `string name` and `float grade`.

The data structure should implement functions to add, remove and search for an element, as shown in the lectures.

Specific Steps for Implementation

1) Your code should be able to receive an input from the command line when executed, and use it to seed the random number generator, as with assignments 1 and 2.
2) Initialize a binary tree and add the following students **in a random order**. To find the position to add the objects, you should use `name` to guide the search/insertion process. As you add a new `student` object, assign a random value to `grade`, in the range [0,100] to each of them.

Adam	Cameron	Jackson	KiSoon	Nicholas
Adrian	Chris	Jacob	Lance	Ryan
Alexander	Damian	James	Liam	Sang
Andrew	David	Jared	Madison	Shane
Ashley	Dillon	Jodi	Magdalena	Simon
Benjamin	Dylan	Jonathan	Marcus	Thomas
Bradley	Ethan	Joshua	Mark	Timothy
Brobie	Frederik	Julius	Melanie	Trent
Callan	Hong	Kelly	Min	Troy
Callum	Hugh	Kenias	Mitchell	Zaanif

3) Print out all names and grades using the inorder traversal, i.e the output should be:

```
(name1, grade1) (name2, grade2) … (nameN, gradeN)
```

4) Print out how many students had a grade ≥ 85, again, by recursively traversing the tree, i.e. the output should be "`HDs: XX`"
5) Print out the average grade of the class, also by recursively traversing the tree, i.e. the output should be "`Average: XX`"
6) Delete all students who had a grade ≤ 50. Print out all names and grades of the remaining students; and the average grade of the class. When a node is deleted, it should be replaced so that the tree that maintains the binary search consistency. *This is a complex procedure and you should refer to the textbook, pp. 1279 to 1286, to understand all the possibilities for node deletion.* We will also talk about them during the classes.
7) Redo steps 3, 4 and 5 with the new tree (after the deletion).

Your submission should be made using the Assignments section of the course Blackboard site. **Incorrectly submitted assignments will not be marked. Assignments that do not use the specified class names will not be further marked.** You should provide all your `.h`, `.cpp` and `.template` files, and a `Makefile`. Also, if necessary, provide a `readme.txt` file containing any instructions for the marker. Each program file should have a proper header section including your name, course and student number, and your code should be properly documented.

Remember that your code should compile and run correctly using Cygwin. There should be no segmentation faults or memory leaks during or after the execution of the program.

Compress all your files, including the cover sheet, into a single .zip file and submit it in by clicking in a link that I will create in the Assignments section on Blackboard especially for that.

Late submissions are subject to the rules specified in the Course Outline. Finally, a completed Assignment Cover Sheet should accompany your submission.

This assignment is worth 15 marks of your final result for the course.